Technical Report 1384
December 1990

BMD/ADA Bit-Oriented Message Definer

A Tool To Define Bit-Oriented Messages in Ada

R. H. Mumm
S. A. Parker

NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

H. R. TALKINGTON, Acting
Technical Director

ADMINISTRATIVE INFORMATION

The work reported in this report was done under the Communications and Networking block program, Independent Exploratory Development (IED) project ZE91, "Automatic Package Specification Generator for Bit-Oriented Messages." The work was done in FY90 by R. H. Mumm and S. A. Parker of the Computer Systems and Software Technology Branch, NOSC, Code 411.

Released by
D. L. Hayward, Head
Computer Systems and
Software Technology
Branch

Under authority of
A. G. Justice, Head
Information Processing
and Displaying Division

ACKNOWLEDGMENTS

The authors wish to acknowledge Dr. Charles Sampson, Computer Science Corporation, and Dr. Michael Shapiro, Naval Ocean Systems Center, for reviewing this document and providing valuable comments.

A special thanks to Michele Kendall, Naval Ocean Systems Center, for graciously doing the word processing required for this document, and to Bob Ollerton, Naval Ocean Systems Center, for technical assistance.

OBJECTIVE

Our objective was to develop a portable software tool that automatically generates the Ada source code to define bit-oriented messages. This tool can help reduce development costs for command and control systems and other Ada software that process bit-oriented messages.

RESULTS

We designed, coded, and demonstrated the Bit-Oriented Message Definer (BMD) tool. The tool compiles and executes in the Vax Ada, Sun Telesoft, and Sun Alsys environments and generates message definitions for five target environments. A preliminary version was used to generate approximately 7000 Ada source lines of code for the Ada Bit-Oriented Message Handler (ABOM) Project.

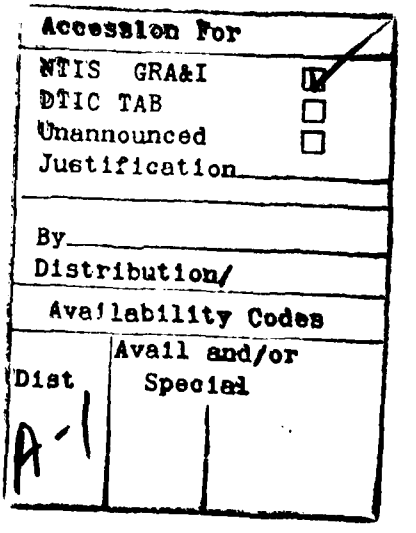
RECOMMENDATIONS

1. Programmers who are developing Ada software that process bit-oriented messages should try using the BMD. BMD can reduce development time.

2. Ada compiler validation testing needs to be more comprehensive. Validation testing currently does not test representation specifications but needs to. During testing of the BMD, problems were found with the representation specifications for two validated compilers.

3. Automatic code generation can be used effectively for Ada applications. The automatic code is most appropriate when the source code is repetitious and writing it manually is labor intensive.

4. Even when writing small programs a design methodology should be used. Small programs frequently expand. A good design facilitates changes, maintenance, and reuse.



CONTENTS

FIGURES

INTRODUCTION

The Bit-Oriented Message Definer (BMD) is a portable interactive software tool for automatically generating Ada source code that defines bit-oriented messages. The BMD can help reduce the development costs for Ada software that processes bit-oriented messages. Appropriate applications include command and control systems, such as, the Command and Control Processor (C^2P) Shadow Project, which is being directed by Naval Ocean Systems Center (NOSC) and the Ship Gridlock Project, which is being conducted at the Naval Surface Warfare Center (NSWC), Dahlgren, VA.

The BMD executes on the Vax Ada, Sun Telesoft Ada, and Sun Alsys compilers and generates source code that will execute in at least five different target environments.

The BMD automatically generates the Ada source code that a programmer would otherwise need to manually generate. This manual process is time consuming and requires specific knowledge of the target computer and compiler. Information the programmer would have to know includes how bits are numbered, how they are ordered, the size of type integer, as well as other details. The BMD user does not need to know these differences. The source code for the message definitions is generated automatically for the target environment the user selects. The BMD user simply defines the messages by looking at pictures of message layouts in software specification documents and then by responding to interactive program prompts. The user needs only a minimal knowledge of Ada.

BACKGROUND

The BMD tool was developed as an Independent Exploratory Development project at NOSC. A need for this tool was identified during the development of software for the Ada Bit-Oriented Message Handler (ABOM) project, carried out by Science Applications International Corporation (SAIC) under the direction of NOSC, Code 411. ABOM software processed hundreds of bit-oriented messages. These included link-11, link-4a, link-16, as well as, Combat Direction System (CDS) messages. Writing the message definitions in Ada by hand was labor intensive, repetitious, and tedious. Errors were easily made. We determined the message definition process could be rapidly and accurately done by automating it. Preliminary BMD work began during the development of ABOM software.

The approach taken to develop BMD was to collect the definitions of bit-oriented messages from actual Ada projects. These projects were

1. AN/WSN-5 Inertial Navigation System (INS) developed by the Software Engineering Institute (SEI)

2. Command and Control Processor (C^2P) Shadow developed by the Hughes Aircraft Corporation and Syscon Corporation for NOSC, Code 411

3. Ada Bit-Oriented Message Handler (ABOM) software developed by SAIC for NOSC, Code 411

The BMD was developed to be general purpose, allowing messages to be defined in different ways. The BMD provides the capability for defining messages required for these projects and other applications.

KEY FEATURES

HOST ENVIRONMENTS

Vax Ada

Sun Telesoft

Sun Alsys

BMD is a very portable tool. BMD will compile and execute in these environments.

TARGET ENVIRONMENTS

Vax Ada

Sun Telesoft Ada

AN-UYK/43 Ada_L

AN-UYK/44 Ada_M

Sun Alsys Ada

CHARACTERISTICS OF THE BMD

a. Provides capability for defining different size messages whose fields vary in length

b. Allows discriminated records to be defined

   This feature is useful when messages have many common fields.

c. Allows multiple messages to be defined in one package

d. Provides capability for assigning default values to fields

   Default values can be assigned in base 2 through 16 using *Ada-like* notation.

e. Allows use of field types defined in global or definitions packages.

f. Does extensive error checking

   Error checking includes (1) examining the validity of file names, (2) checking field names, package names, and type names to determine if they are valid Ada identifiers, (3) checking field position to ensure previously defined fields are not overlaid, and (4) determining if the size of a default integer is too large for the field size. Error messages are displayed to allow the user to make corrections.

g. Writes descriptive header to package specification

   The header gives the user name, output file name, name of target environment, the date and time the package specification was created and other relevant information.

CHARACTERISTICS OF SOURCE CODE GENERATED

a. Defines messages as records

   Record representation clauses are automatically generated to specify the order, position, and size of message fields.

b. Defines message fields to be integer types

   Message fields are defined to be integer types of the appropriate size for the field widths. The integer type statements and length clauses are generated automatically.

c. Aligns records on word boundaries

TOOL LIMITS

a. Maximum number of messages allowed in a package specification: 200

b. Maximum number of bits in a message word: 64

c. Maximum number of bits in a message: 10,000

d. Maximum field size: 32 bits

e. Maximum identifier length: 80 characters

   Note: These limits are declared in package Spec_Gen_.ada

EXAMPLES

The use of BMD will be illustrated with two examples. The first example is the definition of two INS messages. The second illustrates the use of a discriminated record to define three dummy track messages.

INS MESSAGES EXAMPLE

In this example, the Time and Status Message and Attitude Message are defined first for a Vax Ada target environment and then for a Sun Alsys target environment. This example illustrates the difference in the source code generated for these targets.

Figure 1 shows the message layout for the Time and Status Message while figure 2 depicts the layout for the Attitude Message. These are actual message formats taken from the interface design specification document for the INS (NAVSEA, 1982).

Figure 3 shows the BMD prompts and user inputs to generate these message definitions for a Vax Ada target environment. All user inputs are shown in italics. The user, Hans, entered run BMD to begin execution. Hans then entered his name and selected Vax_Ada as the target environment. He indicated the output file name was INS_File, that the number of bits in a message word is 16, the package specification created will be named INS_Spec, the message group name was INS, the message objects will be defined, and the Time_and_Status and Attitude message will be created. The fields in the message were defined

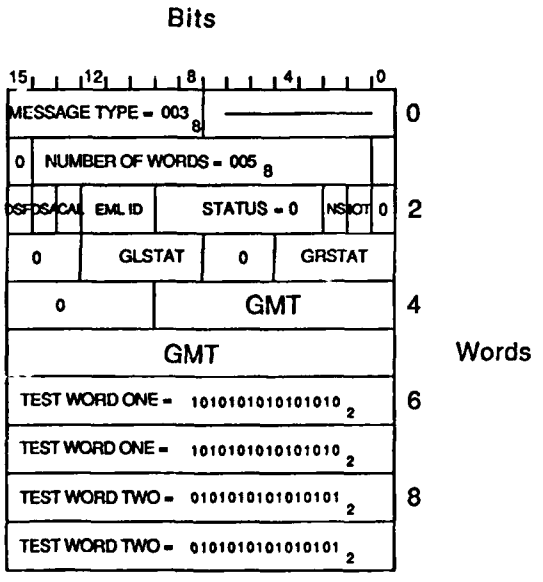


Figure 1. Time and status message for the AN/WSN-5 inertial navigation set.

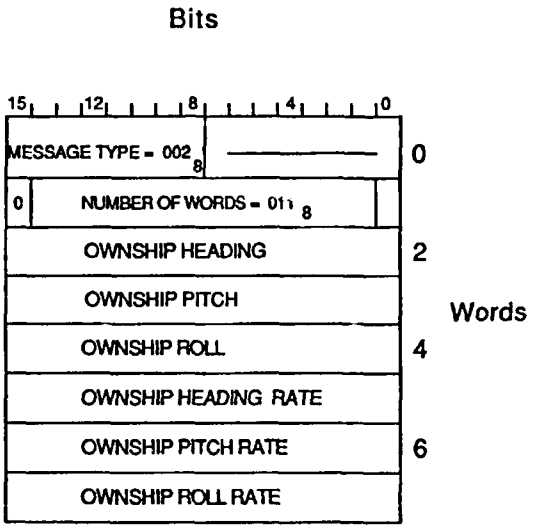


Figure 2. Attitude message for the AN/WSN-5 inertial navigation set.

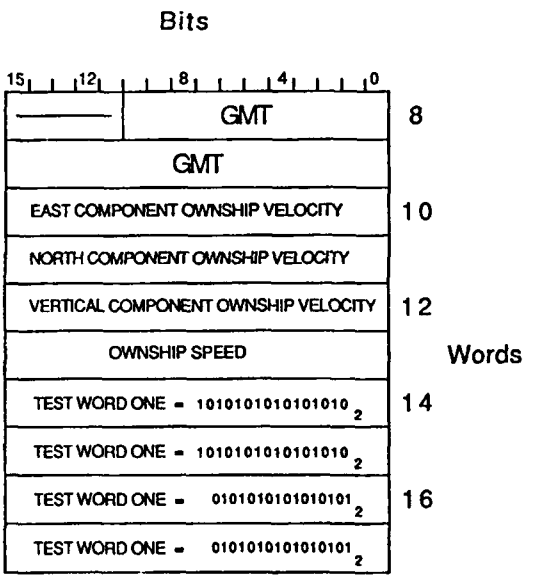


Figure 2. (Continued) Attitude message for the AN/WSN-5 inertial navigation set.

```
run bmd
Enter user name >Hans
Enter target environment
(VAX_ADA,ADA_L,ADA_M,SUN_TEL,SUN_ALSYS)>vax_ada
Enter name of file that will contain specs >INS_File.
Enter number of bits in a message word >16
Enter name of package specification to create >INS_Spec
Enter name of message group >INS
Create message objects (Y or N)? >y
Enter message names in msg group. Enter ';' for last msg.
          1 >Time_and_Status
          2 >Attitude
          3 >;

You are now ready to enter fields for all messages.
Enter ';' for last field.

Is Message Time_and_Status going to use discriminated records
(Y or N)? >n
Enter FIELD_NAME [PACKAGE_NAME.TYPE_NAME] [DEFAULT_VALUE] for
message Time_and_Status
Time_and_Status           1 >Message_Type 8#003#
Enter start word number & start bit position of field (ww,bb) >0,8
Enter end word number & end bit position of field (ww,bb) >0,15
Time_and_Status           2 >Number_of_Words 8#005#
Enter start word number & start bit position of field (ww,bb) >1,1
Enter end word number & end bit position of field (ww,bb) >1,14
Time_and_Status           3 >Zero_Bit1 0
Enter start word number & start bit position of field (ww,bb) >2,0
Enter end word number & end bit position of field (ww,bb) >2,0
Time_and_Status           4 >IOT
Enter start word number & start bit position of field (ww,bb) >2,1
Enter end word number & end bit position of field (ww,bb) >2,1
Time_and_Status           5 >NS
Enter start word number & start bit position of field (ww,bb) >2,2
Enter end word number & end bit position of field (ww,bb) >2,2
Time_and_Status           6 >Status 0
Enter start word number & start bit position of field (ww,bb) >2,3
Enter end word number & end bit position of field (ww,bb) >2,9
Time_and_Status           7 >EML_ID
Enter start word number & start bit position of field (ww,bb) >2,10
Enter end word number & end bit position of field (ww,bb) >2,12
Time_and_Status           8 >CAL
Enter start word number & start bit position of field (ww,bb) >2,13
Enter end word number & end bit position of field (ww,bb) >2,13
Time_and_Status           9 >DSA
Enter start word number & start bit position of field (ww,bb) >2,14
Enter end word number & end bit position of field (ww,bb) >2,14
Time_and_Status          10 >DSF
Enter start word number & start bit position of field (ww,bb) >2,15
Enter end word number & end bit position of field (ww,bb) >2,15
Time_and_Status          11 >GRSTAT
Enter start word number & start bit position of field (ww,bb) >3,0
Enter end word number & end bit position of field (ww,bb) >3,4
Time_and_Status          12 >Zero_Word1 0
Enter start word number & start bit position of field (ww,bb) >3,5
Enter end word number & end bit position of field (ww,bb) >3,7
Time_and_Status          13 >GLSTAT
```

Figure 3. Prompts and user input for time and status and attitude message for Vax Ada target.

```
Enter start word number & start bit position of field (ww,bb) >3,8
Enter end word number & end bit position of field (ww,bb) >3,12
Time_and_Status          14 >Zero_Word2 0
Enter start word number & start bit position of field (ww,bb) >3,13
Enter end word number & end bit position of field (ww,bb) >3,15
Time_and_Status          15 >GMT1
Enter start word number & start bit position of field (ww,bb) >4,0
Enter end word number & end bit position of field (ww,bb) >4,10
Time_and_Status          16 >Zero_Word3 0
Enter start word number & start bit position of field (ww,bb) >4,11
Enter end word number & end bit position of field (ww,bb) >4,15
Time_and_Status          17 >GMT2
Enter start word number & start bit position of field (ww,bb) >5,0
Enter end word number & end bit position of field (ww,bb) >5,15
Time_and_Status          18 >Test_Word1a 2#1010101010101010#
Enter start word number & start bit position of field (ww,bb) >6,0
Enter end word number & end bit position of field (ww,bb) >6,15
Time_and_Status          19 >Test_Word1b 2#1010101010101010#
Enter start word number & start bit position of field (ww,bb) >7,0
Enter end word number & end bit position of field (ww,bb) >7,15
Time_and_Status          20>Test_Word2a 2#0101010101010101#
Enter start word number & start bit position of field (ww,bb) >8,0
Enter end word number & end bit position of field (ww,bb) >8,15
Time_and_Status          21 >Test_Word2b 2#0101010101010101#
Enter start word number & start bit position of field (ww,bb) >9,0
Enter end word number & end bit position of field (ww,bb) >9,15
Time_and_Status          22 >;

Is Message Attitude going to use discriminated records
(Y or N)? >n
Enter FIELD_NAME [PACKAGE_NAME.TYPE_NAME] [DEFAULT_VALUE] for
message Attitude
Attitude            1 >Message_Type 8#002#
Enter start word number & start bit position of field (ww,bb) >0,8
Enter end word number & end bit position of field (ww,bb) >0,15
Attitude            2 >Number_of_Words 8#011#
Enter start word number & start bit position of field (ww,bb) >1,1
Enter end word number & end bit position of field (ww,bb) >1,14
Attitude            3 >Zero_Bit1 0
Enter start word number & start bit position of field (ww,bb) >1,15
Enter end word number & end bit position of field (ww,bb) >1,15
Attitude            4 >Ownship_Heading
Enter start word number & start bit position of field (ww,bb) >2,0
Enter end word number & end bit position of field (ww,bb) >2,15
Attitude            5 >Ownship_Pitch
Enter start word number & start bit position of field (ww,bb) >3,0
Enter end word number & end bit position of field (ww,bb) >3,15
Attitude            6 >Ownship_Roll
Enter start word number & start bit position of field (ww,bb) >4,0
Enter end word number & end bit position of field (ww,bb) >4,15
Attitude            7 >Ownship_Heading_Rate
Enter start word number & start bit position of field (ww,bb) >5,0
Enter end word number & end bit position of field (ww,bb) >5,15
Attitude            8 >Ownship_Pitch_Rate
```

Figure 3. (Continued) Prompts and user input for time and status and attitude message for Vax Ada target.

```
Enter start word number & start bit position of field (ww,bb) >6,0
Enter end word number & end bit position of field (ww,bb) >6,15
Attitude          9 >Ownship_Roll_Rate
Enter start word number & start bit position of field (ww,bb) >7,0
Enter end word number & end bit position of field (ww,bb) >7,15
Attitude         10 >GMT1
Enter start word number & start bit position of field (ww,bb) >8,0
Enter end word number & end bit position of field (ww,bb) >8,10
Attitude         11 >Zero_Word1 0
Enter start word number & start bit position of field (ww,bb) >8,11
Enter end word number & end bit position of field (ww,bb) >8,15
Attitude         12 >GMT2
Enter start word number & start bit position of field (ww,bb) >9,0
Enter end word number & end bit position of field (ww,bb) >9,15
Attitude         13 >East_Comp_Own_Vel
Enter start word number & start bit position of field (ww,bb) >10,0
Enter end word number & end bit position of field (ww,bb) >10,15
Attitude         14 >North_Comp_Own_Vel
Enter start word number & start bit position of field (ww,bb) >11,0
Enter end word number & end bit position of field (ww,bb) >11,15
Attitude         15 >Vert_Comp_Own_Vel
Enter start word number & start bit position of field (ww,bb) >12,0
Enter end word number & end bit position of field (ww,bb) >12,15
Attitude         16 >Ownship_Speed
Enter start word number & start bit position of field (ww,bb) >13,0
Enter end word number & end bit position of field (ww,bb) >13,15
Attitude         17 >Testword1a 2#1010101010101010#
Enter start word number & start bit position of field (ww,bb) >14,0
Enter end word number & end bit position of field (ww,bb) >14,15
Attitude         18 >Testword1b 2#1010101010101010#
Enter start word number & start bit position of field (ww,bb) >15,0
Enter end word number & end bit position of field (ww,bb) >15,15
Attitude         19 >Testword2a 2#0101010101010101#
Enter start word number & start bit position of field (ww,bb) >16,0
Enter end word number & end bit position of field (ww,bb) >16,15
Attitude         20 >Testword2b 2#0101010101010101#
Enter start word number & start bit position of field (ww,bb) >17,0
Enter end word number & end bit position of field (ww,bb)>17,15
Attitude         21 >;
The package specification INS_Spec includes the INS messages below.
Time_and_Status
Attitude
The file you created is :INS_File.
Bit-Oriented Message Definer (BMD) has completed.
```

Figure 3. (Continued) Prompts and user input for time and status and attitude message for Vax Ada target.

by giving them a field name, an optional type defined in another package specification, and an optional field value. If an optional Package_Name.Type_Name is not given, then the field will be of an integer type defined in package INS_Spec. Some fields, such as Message_Type and Number_of_Words were given default values in *Ada-like* notation. Base values of 2 through 16 are valid. Each field position is specified by giving both the field start word, start bit position, and end word, end bit position. To complete the Time and Status message, Hans entered a ";" for FIELD_NAME. In the BMD a ";" is a delimiter indicating there is no more data of the type being entered. Information was entered in similar fashion for the Attitude message. The output file, INS_File, was created and the BMD run completed when ";" was entered for FIELD_NAME for the Attitude message.

Figure 4 depicts the Time and Status and Attitude Message definitions for a Vax Ada target environment. Note the header information containing the user name, file name, target environment, message word size, and time created. Below the header are the type for integer types ranging from 1 bit up through the maximum size of the predefined type integer for the target environment, which is 2**31-1 for Vax Ada. The messages are defined as records. Record representation clauses are used to specify the position and size of message fields.

Figure 5 shows the Time and Status and Attitude Message definitions for a Sun Alsys target environment. The user inputs to generate these definitions were identical to those that generated the Vax Ada definitions, figure 3, except now the user specified that the target was the Sun Alsys. The bit positions relative to the zero storage unit in the record representation clause in figure 5 are different than those for the Vax target shown in figure 4. This difference occurred because the Sun orders bits within a word from left to right whereas the Vax orders bits within a word from right to left.

```
package INS_Spec is

--  User Name: Hans
--  File name is: INS_File.
--  Target environment: VAX_ADA
--  Number of bits in type integer for target environment:  32
--  Number of bits in message word:  16
--  Date package specification created:  8/ 25/ 90
--  Time package specification created:  14: 22: 48
--  The following statements are for INS messages:

 type Integer01 is range 0..1;
 for Integer01'size use 1;
 type Integer02 is range 0..2**2-1;
 for Integer02'size use 2;
 type Integer03 is range 0..2**3-1;
 for Integer03'size use 3;
 type Integer04 is range 0..2**4-1;
 for Integer04'size use 4;
 type Integer05 is range 0..2**5-1;
 for Integer05'size use 5;
 type Integer06 is range 0..2**6-1;
 for Integer06'size use 6;
 type Integer07 is range 0..2**7-1;
 for Integer07'size use 7;
 type Integer08 is range 0..2**8-1;
 for Integer08'size use 8;
 type Integer09 is range 0..2**9-1;
 for Integer09'size use 9;
 type Integer10 is range 0..2**10-1;
 for Integer10'size use 10;
 type Integer11 is range 0..2**11-1;
 for Integer11'size use 11;
 type Integer12 is range 0..2**12-1;
 for Integer12'size use 12;
 type Integer13 is range 0..2**13-1;
 for Integer13'size use 13;
 type Integer14 is range 0..2**14-1;
 for Integer14'size use 14;
 type Integer15 is range 0..2**15-1;
 for Integer15'size use 15;
 type Integer16 is range 0..2**16-1;
 for Integer16'size use 16;

 type Time_and_Status_Record_Type is record
   Message_Type                : Integer08:= 8#003#;
   Number_of_Words             : Integer14 :=8#005#;
   Zero_Bit1                   : Integer01 := 0;
   IOT                         : Integer01;
   NS                          : Integer01;
   Status                      : Integer07;
   EML_ID                      : Integer03 := 0;
   CAL                         : Integer01;
   DSA                         : Integer01;
   DSF                         : Integer01;
   GRSTAT                      : Integer05;
   Zero_Word1                  : Integer03 := 0;
   GLSTAT                      : Integer05;
```

Figure 4. Time and status and attitude message definitions for Vax Ada target.

```
   Test_Word1a                  : Integer16 := 2#10101010101010#;
   Test_Word1b                  : Integer16 := 2#10101010101010#;
   Test_Word2a                  : Integer16 := 2#01010101010101#;
   Test_Word2b                  : Integer16 := 2#01010101010101#;
end record;
for Time_and_Status_Record_Type use record at mod  4;
   Message_Type                 at           0 range  8 .. 15;
   Number_of_Words              at           0 range  17 .. 30;
   Zero_Bit1                    at           0 range  32 .. 32;
   IOT                          at           0 range  33 .. 33;
   NS                           at           0 range  34 .. 34;
   Status                       at           0 range  35 .. 41;
   XML_ID                       at           0 range  42 .. 44;
   CAL                          at           0 range  45 .. 45;
   DSA                          at           0 range  46 .. 46;
   DSF                          at           0 range  47 .. 47;
   GRSTAT                       at           0 range  48 .. 52;
   Zero_Word1                   at           0 range  53 .. 55;
   GLSTAT                       at           0 range  56 .. 60;
   Zero_Word2                   at           0 range  61 .. 63;
   GMT1                         at           0 range  64 .. 74;
   Zero_Word3                   at           0 range  75 .. 79;
   GMT2                         at           0 range  80 .. 95;
   Test_Word1a                  at           0 range  96 .. 111;
   Test_Word1b                  at           0 range  112 .. 127;
   Test_Word2a                  at           0 range  128 .. 143;
   Test_Word2b                  at           0 range  144 .. 159;
end record;
Time_and_Status_Record : Time_and_Status_Record_Type;

type Attitude_Record_Type is record
   Message_Type                 : Integer08 := 8#002#;
   Number_of_Words              : Integer14 := 8#011#;
   Zero_Bit1                    : Integer01 := 0;
   Ownship_Heading              : Integer16;
   Ownship_Pitch                : Integer16;
   Ownship_Roll                 : Integer16;
   Ownship_Heading_Rate         : Integer16;
   Ownship_Pitch_Rate           : Integer16;
   Ownship_Roll_Rate            : Integer16;
   GMT1                         : Integer11;
   Zero_Word1                   : Integer05 := 0;
   GMT2                         : Integer16;
   East_Comp_Own_Vel            : Integer16;
   North_Comp_Own_Vel           : Integer16;
   Vert_Comp_Own_Vel            : Integer16;
   Ownship_Speed                : Integer16;
   Testword1a                   : Integer16 := 2#10101010101010#;
   Testword1b                   : Integer16 := 2#10101010101010#;
   Testword2a                   : Integer16 := 2#01010101010101#;
   Testword2b                   : Integer16 := 2#01010101010101#;
end record;
for Attitude_Record_Type use record at mod  4;
   Message_Type                 at           0 range  8 .. 15;
```

Figure 4. (Continued) Time and status and attitude message definitions for Vax Ada target.

```
    Number_of_Words              at          0 range  17 .. 30;
    Zero_Bit1                    at          0 range  31 .. 31;
    Ownship_Heading              at          0 range  32 .. 47;
    Ownship_Pitch                at          0 range  48 .. 63;
    Ownship_Roll                 at          0 range  64 .. 79;
    Ownship_Heading_Rate         at          0 range  80 .. 95;
    Ownship_Pitch_Rate           at          0 range  96 .. 111;
    Ownship_Roll_Rate            at          0 range  112 .. 127;
    GMT1                         at          0 range  128 .. 138;
    Zero_Word1                   at          0 range  139 .. 143;
    GMT2                         at          0 range  144 .. 159;
    East_Comp_Own_Vel            at          0 range  160 .. 175;
    North_Comp_Own_Vel           at          0 range  176 .. 191;
    Vert_Comp_Own_Vel            at          0 range  192 .. 207;
    Ownship_Speed                at          0 range  208 .. 223;
    Testword1a                   at          0 range  224 .. 239;
    Testword1b                   at          0 range  240 .. 255;
    Testword2a                   at          0 range  256 .. 271;
    Testword2b                   at          0 range  272 .. 287;
  end record;
  Attitude : Attitude_Record_Type;

end INS_Spec;
```

Figure 4. (Continued) Time and status and attitude message definitions for Vax Ada target.

```
package INS_Spec is

--  User Name: hans
--  File name is: INS_Sun.
--  Target environment: SUN_ALSYS
--  Number of bits in type integer for target environment:  16
--  Number of bits in message word:  16
--  Date package specification created:  8/ 27/ 90
--  Time package specification created:  11: 23: 12
--  The following statements are for INS messages:

 type Integer01 is range 0..1;
 for Integer01'size use 1;
 type Integer02 is range 0..2**2-1;
 for Integer02'size use 2;
 type Integer03 is range 0..2**3-1;
 for Integer03'size use 3;
 type Integer04 is range 0..2**4-1;
 for Integer04'size use 4;
 type Integer05 is range 0..2**5-1;
 for Integer05'size use 5;
 type Integer06 is range 0..2**6-1;
 for Integer06'size use 6;
 type Integer07 is range 0..2**7-1;
 for Integer07'size use 7;
 type Integer08 is range 0..2**8-1;
 for Integer08'size use 8;
 type Integer09 is range 0..2**9-1;
 for Integer09'size use 9;
 type Integer10 is range 0..2**10-1;
 for Integer10'size use 10;
 type Integer11 is range 0..2**11-1;
 for Integer11'size use 11;
 type Integer12 is range 0..2**12-1;
 for Integer12'size use 12;
 type Integer13 is range 0..2**13-1;
 for Integer13'size use 13;
 type Integer14 is range 0..2**14-1;
 for Integer14'size use 14;
 type Integer15 is range 0..2**15-1;
 for Integer15'size use 15;
 type Integer16 is range 0..2**16-1;
 for Integer16'size use 16;


 type Time_and_Status_Record_Type is record
   Message_Type              : Integer08 := 8#003#;
   Number_of_Words           : Integer14 := 8#005#;
   Zero_Bit1                 : Integer01 := 0;
   IOT                       : Integer01;
   NS                        : Integer01;
   Status                    : Integer07 := 0;
   EML_ID                    : Integer03;
   CAL                       : Integer01;
   DSA                       : Integer01;
   DSF                       : Integer01;
   GRSTAT                    : Integer05;
```

Figure 5. Time and status and attitude message definitions for Sun Alsys target.

```
   Zero_Word1                  : Integer03 := 0;
   CLSTAT                      : Integer05;
   Zero_Word2                  : Integer03 := 0;
   GMT1                        : Integer11;
   Zero_Word3                  : Integer05 := 0;
   GMT2                        : Integer16;
   Test_Word1a                 : Integer16 := 2#10101010101010#;
   Test_Word1b                 : Integer16 := 2#10101010101010#;
   Test_Word2a                 : Integer16 := 2#01010101010101#;
   Test_Word2b                 : Integer16 := 2#01010101010101#;
end record;
for Time_and_Status_Record_Type use record at mod  2;
   Message_Type                at          0 range  0 .. 7;
   Number_of_Words             at          0 range  17 .. 30;
   Zero_Bit1                   at          0 range  47 .. 47;
   IOT                         at          0 range  46 .. 46;
   NS                          at          0 range  45 .. 45;
   Status                      at          0 range  38 .. 44;
   EML_ID                      at          0 range  35 .. 37;
   CAL                         at          0 range  34 .. 34;
   DSA                         at          0 range  33 .. 33;
   DSF                         at          0 range  32 .. 32;
   CRSTAT                      at          0 range  59 .. 63;
   Zero_Word1                  at          0 range  56 .. 58;
   GLSTAT                      at          0 range  51 .. 55;
   Zero_Word2                  at          0 range  48 .. 50;
   GMT1                        at          0 range  69 .. 79;
   Zero_Word3                  at          0 range  64 .. 68;
   GMT2                        at          0 range  80 .. 95;
   Test_Word1a                 at          0 range  96 .. 111;
   Test_Word1b                 at          0 range  112 .. 127;
   Test_Word2a                 at          0 range  128 .. 143;
   Test_Word2b                 at          0 range  144 .. 159;
end record;
Time_and_Status : Time_and_Status_Record_Type;

type Attitude_Record_Type is record
   Message_Type                : Integer08 := 8#002#;
   Number_of_Words             : Integer14 := 8#011#;
   Zero_Bit1                   : Integer01 := 0;
   Ownship_Heading             : Integer16;
   Ownship_Pitch               : Integer16;
   Ownship_Roll                : Integer16;
   Ownship_Heading_Rate        : Integer16;
   Ownship_Pitch_Rate          : Integer16;
   Ownship_Roll_Rate           : Integer16;
   GMT1                        : Integer11;
   Zero_Word1                  : Integer05 := 0;
   GMT2                        : Integer16;
   East_Comp_Own_Vel           : Integer16;
   North_Comp_Own_Vel          : Integer16;
   Vert_Comp_Own_Vel           : Integer16;
   Ownship_Speed               : Integer16;
   Test_Word1a                 : Integer16 := 2#10101010101010#;
   Test_Word1b                 : Integer16 := 2#10101010101010#;
```

Figure 5. (Continued) Time and status and attitude message definitions for Sun Alsys target.

```
   TestWord2a                : Integer16 := 2#01010101010101#;
   TestWord2b                : Integer16 := 2#01010101010101#;
 end record;
 for Attitude_Record_Type use record at mod  2;
   Message_Type              at            0 range  0 .. 7;
   Number_of_Words           at            0 range  17 .. 30;
   Zero_Bit1                 at            0 range  16 .. 16;
   Ownship_Heading           at            0 range  32 .. 47;
   Ownship_Pitch             at            0 range  48 .. 63;
   Ownship_Roll              at            0 range  64 .. 79;
   Ownship_Heading_Rate      at            0 range  80 .. 95;
   Ownship_Pitch_Rate        at            0 range  96 .. 111;
   Ownship_Roll_Rate         at            0 range  112 .. 127;
   GMT1                      at            0 range  133 .. 143;
   Zero_Word1                at            0 range  128 .. 132;
   GMT2                      at            0 range  144 .. 159;
   East_Comp_Own_Vel         at            0 range  160 .. 175;
   North_Comp_Own_Vel        at            0 range  176 .. 191;
   Vert_Comp_Own_Vel         at            0 range  192 .. 207;
   Ownship_Speed             at            0 range  208 .. 223;
   Test_Word1a               at            0 range  224 .. 239;
   Test_Word1b               at            0 range  240 .. 255;
   TestWord2a                at            0 range  256 .. 271;
   TestWord2b                at            0 range  272 .. 287;
 end record;
 Attitude : Attitude_Record_Type;

end INS_Spec;
```

Figure 5. (Continued) Time and status and attitude message definitions for Sun Alsys target.

DUMMY TRACK MESSAGES EXAMPLE

Figures 6, 7, and 8 show the message layout for the Dummy Air Track Message, Dummy Surface Track Message, and Dummy Subsurface Track Message. These are fictitious messages created for illustration purposes only. In these messages the fields in each are identical down through CAL CONST. Beyond CAL CONST the fields are different. This layout suggests that these messages can be defined as one discriminated record.

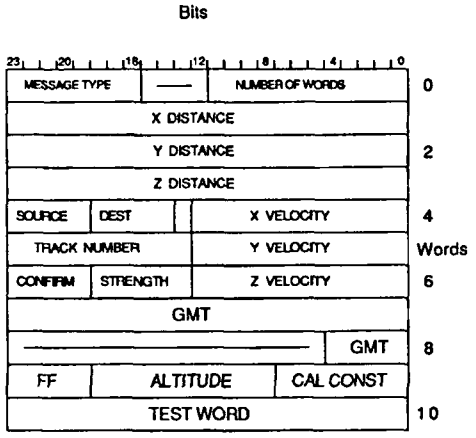


Figure 6. Dummy air track message.

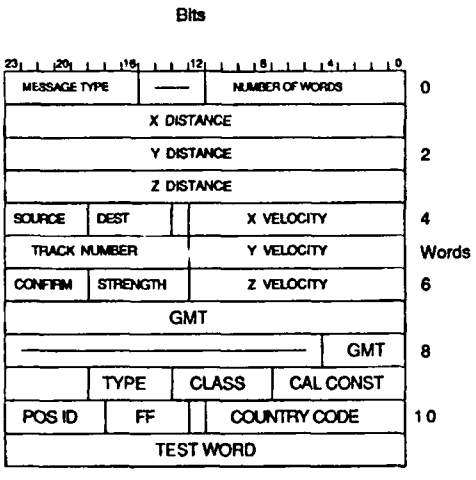


Figure 7. Dummy surface track message.

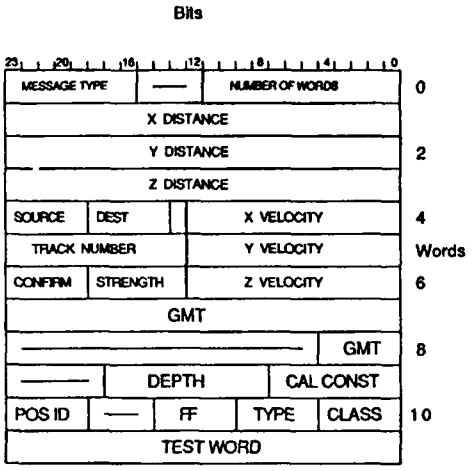


Figure 8. Dummy subsurface track message.

Figure 9 shows the BMD prompts and user inputs used to generate the dummy track message definitions. Again, all user inputs are given in italics. In this example the user, Sally, selected the UYK_43 as the target, indicates that message size is 24 bits and that message objects will not be created. Sally's responses indicate the message name is Track and will be defined as a discriminated record. She defined the discriminant as Msg. The user has three choices when defining the discriminant type. It can be (1) of type Boolean, (2) of a type defined in another package specification or (3) of a type defined by the user. When it is defined by the user, the type definition will be declared in the package created. In figure 9, Sally defined the discriminant type as Global_Data. Message_Type. Next, the fields, Number_of_ Words through Cal_Const are defined to be common to all values of the discriminant. Then Sally indicated the first value of the discriminant is Air, and message fields for the discriminant value Air are Altitude, FF_A, and TestWord_A. The same process was followed for the discriminant values Surface and Sub_Surface. When Sally entered a ";" to indicate there were no more values for the discriminant, the BMD created the output file, track_file., and the BMD terminated. The fields, Message_Type, FF_A, TestWord_A, FF_S, TestWord_S, FF_SS, and TestWord_SS were defined to be of types defined in package Global_Data.

```
run bmd
Enter user name >sally
Enter target environment (VAX_ADA,ADA_L,ADA_M,SUN_TEL,SUN_ALSYS)
>Ada_L
Enter name of file that will contain specs >track_file.
Enter number of bits in a message word >24
Enter name of package specification to create >Track_Message
Enter name of message group >Track
Create message objects (Y or N)? >n
Enter message names in msg group. Enter ';' for last msg.
          1 >Track
          2 >;

You are now ready to enter fields for all messages.
Enter ';' for last field.

Is Message Track going to use discriminated records (Y or N)? >y
What is the Discriminate's name? >Msg

Enter type (Boolean | Package_Name.Type_Name | Type Name) of discriminant
>Global_Data.Message_Type
Enter start word number and start bit position for field, Msg > 0,16
Enter end word number and end bit position for field, Msg > 0,23
Does message Track contain fields common to all values of
discriminate (Y or N)? >y
Enter field information for fields common to all values of Msg
Enter ';' to end definition of common fields
Enter FIELD_NAME [PACKAGE_NAME.TYPE_NAME] [DEFAULT_VALUE]
Track                  1 >Number_of_Words
Enter start word number & start bit position of field (ww,bb) >0,0
Enter end word number & end bit position of field (ww,bb) >0,11
Track                  4 >X_Distance
Enter start word number & start bit position of field (ww,bb) >1,0
Enter end word number & end bit position of field (ww,bb) >1,23
Track                  5 >Y_Distance
Enter start word number & start bit position of field (ww,bb) >2,0
Enter end word number & end bit position of field (ww,bb) >2,23
Track                  6 >Z_Distance
Enter start word number & start bit position of field (ww,bb) >3,0
Enter end word number & end bit position of field (ww,bb) >3,23
Track                  7 >X_Velocity
Enter start word number & start bit position of field (ww,bb) >4,0
Enter end word number & end bit position of field (ww,bb) >4,12
Track                  8 >Dest
Enter start word number & start bit position of field (ww,bb) >4,14
Enter end word number & end bit position of field (ww,bb) >4,18
Track                  9 >Source
Enter start word number & start bit position of field (ww,bb) >4,19
Enter end word number & end bit position of field (ww,bb) >4,23
Track                 10 >Y_Velocity
Enter start word number & start bit position of field (ww,bb) >5,0
Enter end word number & end bit position of field (ww,bb) >5,12
Track                 11 >Track_Number
Enter start word number & start bit position of field (ww,bb) >5,13
Enter end word number & end bit position of field (ww,bb) >5,23
```

Figure 9. Prompts and user input for dummy track messages for AN/UYK-43 Ada_L target.

```
Track                 12 >X_Velocity
Enter start word number & start bit position of field (ww,bb) >6,0
Enter end word number & end bit position of field (ww,bb) >6,12
Track                 13 >Strength
Enter start word number & start bit position of field (ww,bb) >6,13
Enter end word number & end bit position of field (ww,bb) >6,18
Track                 14 >Confirmation
Enter start word number & start bit position of field (ww,bb) >6,19
Enter end word number & end bit position of field (ww,bb) >6,23
Track                 15 >GMT1
Enter start word number & start bit position of field (ww,bb) >7,0
Enter end word number & end bit position of field (ww,bb) >7,23
Track                 16 >GMT2
Enter start word number & start bit position of field (ww,bb) >8,0
Enter end word number & end bit position of field (ww,bb) >8,5
Track                 17 >Cal_Const
Enter start word number & start bit position of field (ww,bb) >9,0
Enter end word number & end bit position of field (ww,bb) >9,7
Track                 17 >;

Enter possible value taken by Msg
Enter ';' if there are no more possible values
>Global_Data.Air
Enter field information needed for this value of discriminate
Enter ';' to end definition of fields for this value
Enter FIELD_NAME [PACKAGE_NAME.TYPE_NAME] [DEFAULT_VALUE]
Track                  1 >Altitude
Enter start word number & start bit position of field (ww,bb) >9,8
Enter end word number & end bit position of field (ww,bb) >9,18
Track                  2 >FF_A Global_Data.FF_Type
Enter start word number & start bit position of field (ww,bb) >9,19
Enter end word number & end bit position of field (ww,bb) >9,23
Track                  3 >TestWord_A Global_Data.TestWord_Type 8#77777777#
Enter start word number & start bit position of field (ww,bb) >10,0
Enter end word number & end bit position of field (ww,bb) >10,15
Track                  4 >;
Enter possible value taken by Msg
Enter ';' if there are no more possible values
Global_Data.Surface
Enter field information needed for this value of discriminate
Enter ';' to end definition of fields for this value
Enter FIELD_NAME [PACKAGE_NAME.TYPE_NAME] [DEFAULT_VALUE]
Track                  1 >Class_A
Enter start word number & start bit position of field (ww,bb) >9,8
Enter end word number & end bit position of field (ww,bb) >9,13
Track                  2 >Type_S
Enter start word number & start bit position of field (ww,bb) >9,14
Enter end word number & end bit position of field (ww,bb) >9,18
Track                  3 >Country_Code
Enter start word number & start bit position of field (ww,bb) >10,0
Enter end word number & end bit position of field (ww,bb) >10,11
Track                  4 >FF_S Global_Data.FF_Type
```

Figure 9. (Continued) Prompts and user input for dummy track messages for AN/UYK-43 Ada_L target.

```
Enter start word number & start bit position of field (ww,bb) >10,13
Enter end word number & end bit position of field (ww,bb) >10,17
Track                  5 >Pos_ID_S
Enter start word number & start bit position of field (ww,bb) >10,18
Track                  3 >Country_Code
Enter end word number & end bit position of field (ww,bb) >10,23
Track                  6 >Testword_S Global_Data.TestWord_Type 8#77777777#
Enter start word number & start bit position of field (ww,bb) >11,0
Enter end word number & end bit position of field (ww,bb) >11,23
Track                  7 >;
Enter possible value taken by Msg
Enter ';' if there are no more possible values
>Global_Data.Sub_Surface
Enter field information needed for this value of discriminate
Enter ';' to end definition of fields for this value
Enter FIELD_NAME [PACKAGE_NAME.TYPE_NAME] [DEFAULT_VALUE]
Track                  1 >Depth
Enter start word number & start bit position of field (ww,bb) >9,9
Enter end word number & end bit position of field (ww,bb) >9,17
Track                  2 >Class_SS
Enter start word number & start bit position of field (ww,bb) >10,0
Enter end word number & end bit position of field (ww,bb) >10,4
Track                  3 >Type_SS
Enter start word number & start bit position of field (ww,bb) >10,5
Enter end word number & end bit position of field (ww,bb) >10,9
Track                  4 >NT_SS Global_Data.NT_Type
Enter start word number & start bit position of field (ww,bb) >10,10
Enter end word number & end bit position of field (ww,bb) >10,14
Track                  5 >Pos_ID_SS
Enter start word number & start bit position of field (ww,bb) >10,19
Enter end word number & end bit position of field (ww,bb) >10,23
Track                  6 >Test_Word_SS Global_Data.Testword_Type 8#77777777#
Enter start word number & start bit position of field (ww,bb) >11,0
Enter end word number & end bit position of field (ww,bb) >11,15
Track                  7 >;
Enter possible value taken by Msg
Enter ';' if there are no more possible values
;
The package specification Track_Message includes the
Track messages below.
Track
The file you created is :track_file.
Bit-Oriented Message Definer (BMD) has completed.
```

Figure 9. (Continued) Prompts and user input for dummy track messages for AN/UYK-43 Ada_L target.

```
package Global_Data is
  type Message_Type is (Air, Surface, Sub_Surface);
  for Message_Type use (Air => 1, Surface => 2,
                        Sub_Surface => 3);
  for Message_Type'Size use 8;
  type FF_Type is (Friend, Foe);
  for FF_Type use (Friend => 1, Foe => 2);
  for FF_Type'Size use 5;
  type Testword_Type is range 0..16;
  for Testword_Type'Size use 16;
end Global_Data;
```

Figure 10. Package global data for dummy track messages.

Figure 10 shows package Global_Data, which contains type definitions for global message data used by the package created above. Package Global_Data was created by hand.

Figure 11 depicts the discriminated record message definition for the Dummy Air, Dummy Surface, and Dummy Subsurface Messages. This figure contains the source code generated for the BMD prompts and user inputs shown in figure 9.

```
with Global_Data;
package Track_Message is

--  User Name: sally
--  File name is: track_file.
--  Target environment: ADA_L
--  Number of bits in type integer for target environment:  32
--  Number of bits in message word:  24
--  Date package specification created:  8/ 29/ 90
--  Time package specification created:  16: 30: 55
--  The following statements are for Track messages:

type Integer01 is range 0..1;
for Integer01'size use 1;
type Integer02 is range 0..2**2-1;
for Integer02'size use 2;
type Integer03 is range 0..2**3-1;
for Integer03'size use 3;
type Integer04 is range 0..2**4-1;
for Integer04'size use 4;
type Integer05 is range 0..2**5-1;
for Integer05'size use 5;
type Integer06 is range 0..2**6-1;
for Integer06'size use 6;
type Integer07 is range 0..2**7-1;
for Integer07'size use 7;
type Integer08 is range 0..2**8-1;
for Integer08'size use 8;
type Integer09 is range 0..2**9-1;
for Integer09'size use 9;
type Integer10 is range 0..2**10-1;
for Integer10'size use 10;
type Integer11 is range 0..2**11-1;
for Integer11'size use 11;
type Integer12 is range 0..2**12-1;
for Integer12'size use 12;
type Integer13 is range 0..2**13-1;
for Integer13'size use 13;
type Integer14 is range 0..2**14-1;
for Integer14'size use 14;
type Integer15 is range 0..2**15-1;
for Integer15'size use 15;
type Integer16 is range 0..2**16-1;
for Integer16'size use 16;
type Integer17 is range 0..2**17-1;
for Integer17'size use 17;
type Integer18 is range 0..2**18-1;
for Integer18'size use 18;
type Integer19 is range 0..2**19-1;
for Integer19'size use 19;
type Integer20 is range 0..2**20-1;
for Integer20'size use 20;
type Integer21 is range 0..2**21-1;
for Integer21'size use 21;
type Integer22 is range 0..2**22-1;
for Integer22'size use 22;
```

Figure 11. Dummy track message definitions for AN/UYK-43 Ada_L target.

```
type Integer23 is range 0..2**23-1;
for Integer23'size use 23;
type Integer24 is range 0..2**24-1;
for Integer24'size use 24;
type Integer25 is range 0..2**25-1;
for Integer25'size use 25;
type Integer26 is range 0..2**26-1;
for Integer26'size use 26;
type Integer27 is range 0..2**27-1;
for Integer27'size use 27;
type Integer28 is range 0..2**28-1;
for Integer28'size use 28;
type Integer29 is range 0..2**29-1;
for Integer29'size use 29;
type Integer30 is range 0..2**30-1;
for Integer30'size use 30;
type Integer31 is range 0..2**31-1;
for Integer31'size use 31;
type Integer32 is range -2**31..2**31-1;
for Integer32'size use 32;

type Track_Message_Record_Type (Msg : Global_Data.Message_Type) is
  record
    Number_of_Words             : Integer12;
    X_Distance                  : integer24;
    Y_Distance                  : integer24;
    Z_Distance                  : integer24;
    X_Velocity                  : integer13;
    Dest                        : integer05;
    Source                      : integer05;
    Y_Velocity                  : integer13;
    Track_Number                : integer11;
    Z_Velocity                  : integer13;
    Strength                    : integer06;
    Confirmation                : integer05;
    GMT1                        : integer24;
    GMT2                        : integer06;
    Cal_Const                   : integer08;
    case Msg is
      when Global_Data.Air =>
        Altitude                : Integer11;
        FF_A                    : Global_Data.FF_Type;
        TestWord_A              : Global_Data.TestWord_Type :=
                                  8#77777777#;
      when Global_Data.Surface =>
        Class_S                 : Integer06;
        Type_S                  : Integer05;
        Country_Code            : Integer12;
        FF_S                    : Global_Data.FF_Type;
        Pos_ID_S                : Global_Data.FF_Type;
        Testword_S              : Global_Data.TestWord_Type :=
                                  8#77777777#;
      when Global_Data.Sub_Surface =>
        Depth                   : Integer09;
        Class_SS                : Integer05;
```

Figure 11. (Continued) Dummy track message definitions for AN/UYK-43 Ada_L target.

```
        Type_SS                 : Integer05;
        FF_SS                   : Global_Data.FF_Type;
        Pos_ID_SS               : Global_Data.FF_Type;
        Test_Word_SS            : Global_Data.Testword_Type :=
                                  8#77777777#;

    end case;
  end record;
for Track_Message_Record_Type use record at mod  1;
  Number_of_Words               at          0 range  0 .. 11;
  Msg                                       0 range  16 .. 23;
  X_Distance                    at          0 range  24 .. 47;
  Y_Distance                    at          0 range  48 .. 71;
  Z_Distance                    at          0 range  72 .. 95;
  X_Velocity                    at          0 range  96 .. 108;
  Dest                          at          0 range  110 .. 114;
  Source                        at          0 range  115 .. 119;
  Y_Velocity                    at          0 range  120 .. 132;
  Track_Number                  at          0 range  133 .. 143;
  Z_Velocity                    at          0 range  144 .. 156;
  Strength                      at          0 range  157 .. 162;
  Confirmation                  at          0 range  163 .. 167;
  CMT1                          at          0 range  168 .. 191;
  CMT2                          at          0 range  192 .. 197;
  Cal_Const                     at          0 range  216 .. 223;
  Altitude                      at          0 range  224 .. 234;
  FF_A                          at          0 range  235 .. 239;
  TestWord_A                    at          0 range  240 .. 255;
  Class_S                       at          0 range  224 .. 229;
  Type_S                        at          0 range  230 .. 234;
  Country_Code                  at          0 range  240 .. 251;
  FF_S                          at          0 range  253 .. 257;
  Pos_ID_S                      at          0 range  258 .. 263;
  Testword_S                    at          0 range  264 .. 287;
  Depth                         at          0 range  225 .. 233;
  Class_SS                      at          0 range  240 .. 244;
  Type_SS                       at          0 range  245 .. 249;
  FF_SS                         at          0 range  250 .. 254;
  Pos_ID_SS                     at          0 range  259 .. 263;
  Test_Word_SS                  at          0 range  264 .. 279;
 end record;

end Track_Message;
```

Figure 11. (Continued) Dummy track message definitions for AN/UYK-43 Ada_L target.

REFERENCE

Naval Sea Systems Command. 1982. "Interface Design Specification Inertial Navigation Set AN/WSN-5 to External Computer for Parallel Channels B and C," NAVSEA T9427-AA-IDS-010/WSN-5.

BIBLIOGRAPHY

Management Assistance Corporation of America (MACA). 1989. "Ada Software (ASR) Master Index."

Mumm, H., and R. Ollerton. 1990. "User's Guide to an Event-Activation Record Approach to Simulation Modeling in Ada." NOSC TD 1944. Naval Ocean Systems Center, San Diego, CA.

Myers, B., and A. Cappellini. 1987. "The Use of Representation Clauses and Implementation-Dependent Features in Ada: I. Overview." CMU/SEI-87-TR-14, Software Engineering Institute.

Telesoft. 1989. "The Telesoft Second Generation Ada Development System, TeleGen2 for Sun386i/UNIX, User Guide," CVR-1425N- V1.1(386i/UNIX).

ABBREVIATIONS AND ACRONYMS

ABOM	Ada Bit-Oriented Message Handler
BMD	Bit-Oriented Message Defined
CDS	Combat Direction System
C^2P	Command and Control Processor
INS	AN/WSN-5 Inertial Navigation System
LRM	Language Reference Manual
NASTEE	Network Architecture Simulated Test and Evaluation Environment
NOSC	Naval Ocean Systems Center
NSWC	Naval Surface Warfare Center
SAIC	Science Applications International Corporation
SEI	Software Engineering Institute

APPENDIX A: LESSONS LEARNED

Lessons were learned in a number of areas during the development, testing, and use of the BMD.

1. Reuse

Existing software contributed to the development of the BMD in two ways. First, the Network Architecture Simulated Test and Evaluation Environment (NASTEE) software developed by NOSC, Code 854, provided useful examples of how automatic code generation is done in Ada.

Second, source code was reused from other projects. Code reused included a linked list package from the Event-Activation Record Approach to Simulation Modeling in Ada libraries developed by NOSC, Code 854, and several subprograms for manipulating character strings from the INS developed by the SEI.

The reuse of existing code resulted in faster BMD development with better written components.

The source lines of code for the BMD and source lines of code reused is summarized below.

	Delimiting Semicolon Statements	Comments	Delimiting Semicolon Stmts & Comments	Card-Image Lines
BMD	1243	344	1587	3380
reused	216	151	367	657

2. Tools

The most helpful tools for this project (other than the compilers) were the Vax Ada Debugger and the NASA/Goddard Space Flight Center Standard Pretty Printer. The pretty printer was extracted from the Ada Software Repository (ASR) at White Sands Missile Range, New Mexico. A debugger was not used for development on the Sun but was used for the Vax; the Vax debugging was significantly faster. The pretty printer was helpful in making the style of the Ada code written by the two authors consistent and in improving the code readability.

3. Portability

One goal of the project from the beginning was that the BMD would be 100 percent transportable across the Vax Ada and Sun Telesoft environments, that is, the same software would run in each environment without changing one line of source code. This goal was achieved. Care was taken to not use any software that would limit the BMD to Vax/VMS or Sun/Unix. The BMD has also been successfully compiled and executed in a Sun Alsys environment.

Two potential obstacles to portability were the difference in file names for VMS versus Unix and the difference in the size of the type integer for the two compilers. Type integer is 32 bits for Vax Ada and 16 bits for Sun Telesoft. The difference in file names was handled by examining the contents of System.System_Name to determine the compiler/operating system BMD was running on.

The difference in the size of type integer for the compilers was handled by declaring the integer type below. The predefined integer type was not used.

```
type Interger_Type is range 0..2147483647;
```

The upper bound on the range is 2**31-1. The lower bound is zero because the BMD does not use any negative integers. The Sun Telesoft and Sun Alsys compilers automatically make Integer_Type of type long_integer.

Declaring your own type for integer rather than using the predefined integer type is one portability guideline in Nissen, 1988. This guideline and others in Nissen, 1988 were followed. If this guideline had not been followed then the BMD would not have ported from the Vax Ada to the Sun Telesoft environment.

Portability was also ensured by the sequence of demonstrations, that is, there was one on a Vax, followed by a Sun demonstration, and then one on a Vax again.

4. Testing

Testing was done by first compiling message definition code generated by the BMD. Then a test program was run that printed out the values of the message fields using the message definition generated by the BMD. This testing was done for each of the target environments. The source code generated for each of the five targets compiled with no errors.

Executing the test program in each target environment has been less successful. The test program executed successfully in the Vax Ada and Sun Alsys environments. It did not execute properly with the Sun Telesoft and ALS/N compilers. The problem with these compilers appears to be the record representation clauses do not work properly.

The test program was run using the Telesoft Version 1.1 compiler for a Sun 386I computer. When using the Telesoft compiler the alignment of fields within a message were displaced 1 byte. A trouble report was submitted to Telesoft's customer service group. A Telesoft representative ran the test program on both the Sun 386I Telesoft and Sun 4 Telesoft compilers. She said the alignment problem is a compiler bug that exists with both compilers and that Telesoft is fixing it.

For the ALS/N compiler, the test program was executed on the AdaVax compiler. A constraint error was raised when the test program attempted to print out a 16-bit test word. An error report has been submitted to the ALS/N program office. Testing involving the execution of Ada_L
and Ada_M generated object code on the AN/UYK-43 and AN/UYK-44 computers is currently being done.

A lesson learned during testing of the BMD is validation testing for Ada compilers needs to be more comprehensive. The compilers for the five target environments all passed validation. However, three have problems related to record representation specifications. Currently, record representation specifications and other compiler dependent features covered in Chapter 13 of the Language Reference Manual (LRM), (United States Department of Defense, 1983), are omitted from validation testing. These features must be tested during validation to keep erroneous compilers from going to the field.

5. Timing Comparisons - Manual versus BMD

A preliminary version of the BMD was used by SAIC to define bit-oriented messages for the ABOM project. SAIC manually wrote the source code to define link-11, link-4a, and link-16 messages. The CDS message definitions were generated using the BMD. SAIC reported that message definitions were created twice as fast when using the automated tool. SAIC used the tool to generate approximately 7000 lines of source code defining the CDS messages.

One of the authors, (HM), manually wrote the message definitions for four INS messages targeted for the AN/UYK-43 and then generated them using the BMD. The task required 2 hours and 14 minutes to do manually and 25 minutes to do with the BMD. After writing the code by hand, three compilations were required to fix errors. The BMD-generated code compiled error-free the first time.

6. Lack of a Formal Design Methodology

No formal design methodology was used in developing the BMD. At the beginning it did not seem necessary. It appeared that the tool would remain small and uncomplicated. However, it grew in size and complexity. Ultimately, the use of a formal design methodology would have made it easier for the authors to change and enhance the code.

7. Lack of a Configuration Management Tool

No configuration management tool was used during the development of the BMD. At the beginning it did not seem necessary since only two people would be writing the tool. In hindsight, a configuration management tool would have been useful.

REFERENCES

Nissen, J., and P. Wallis. 1988. *Portability and Style*, Cambridge University Press, Cambridge, England.

United States Department of Defense. 1983. "Reference Manual for the Ada Programming Language," ANSI/MIL-STD-1815A, Honeywell, Minneapolis, MN.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1990	Final: Oct 89 — Sep 90

4. TITLE AND SUBTITLE
BMD/ADA BIT-ORIENTED MESSAGE DEFINER
A Tool to Define Bit-Oriented Messages in Ada

5. FUNDING NUMBERS
PR: ZE91
WU: DN3000040
PE: 0602936N

6. AUTHOR(S)
R. H. Mumm and S. A. Parker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152–5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NOSC TR 1384

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Chief of Naval Research
Independent Exploratory Development Programs (IED)
OCNR-20T
Arlington, VA 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Bit-Oriented Message Definer (BMD) is a portable interactive software tool for automatically generating Ada source code that defines bit-oriented messages. The BMD can help reduce the development costs for Ada software that process bit-oriented messages.

14. SUBJECT TERMS
Ada

15. NUMBER OF PAGES
32

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 0144	R. November	(1)
Code 402	B. Wasilausky	(1)
Code 41	A. Justice	(1)
Code 411	D. Hayward	(1)
Code 411	H. Mumm	(20)
Code 952B	J. Puleo	(1)
Code 961	Archive/Stock	(6)
Code 964B	Library	(3)

Defense Technical Information Center
Alexandria, VA 22304-6145 (4)

NOSC Liaison Office
Washington, DC 20363-5100 (1)

Center for Naval Analyses
Alexandria, VA 22302-0268 (1)

Naval Surface Warfare Center
Dahlgren, VA 22448-5000 (1)

Headquarters, CECOM
AMSEL-RD-AIN-TP
Fort Monmouth, NJ 07703 (1)

UNISYS
St Paul, MN 55164 (1)

SAIC COMSYSTEMS
San Diego, CA 92121 (1)